プログラムに出現する名前の性質について
2R-6　— 名前管理システムの実現に向けて

神田(こうだ)陽治　田中英彦　元岡達
東京大学　工学部

1. 研究の位置付け

我々はORAGA計画と称する「ソフトウェア生産性向上」をコンピュータアーキテクチャのレベルから抜本的に解決することを狙う研究を進めている。ソフトウェア工学を生むほどの大目標に加え，ハードウェアからユーザインタフェスに至る広範な課題設定は，我々の解決能力を越える所にあるかもしれないが各課題で提案するアプローチは独自性があると信じている[1][2]。

4つの課題に大分した。ハードウェアアーキテクチャのOrasa-itself，並列オブジェクト言語のDinnerBell，ユーザインターフェスのObjectPeeper，及びMessageCollector，最後に名前管理システムNameMasterである。

2. SymbolicName　— 識別子の拡張であるObject記述子

Objectを一意に指す内部名がName，Nameを一意に決める外部名がSymbolic Nameである。SymbolicNameはそれが指すObjectに関する情報を集約している。その一部が目に見える識別子として表示される。識別子は固定ではなくその場の状況に応じてSymbolicNameから計算される。開発中のDinnerBellはビットマップエディタの使用を前提に、識別子と見たい情報をマウスで選択すればウィンドウが開いて，元々のSymbolicName内の情報を読み出すことができる。例えば識別子PrintDeviceBusy，「発声」を選択すると［print divάis bίzi］と音声返答する。*

3. 名前付けの必要性　— 的確かつ体系的な名前は理解を助ける

識別子を適切に付けて読み手に指す対象の意味を伝えるように配慮することは，プログラミングの初歩的な約束事である。適切な名前は対象を的確に想起させ、対象間の相違を体系的に見せてくれる。しかし，的確かつ体系的な名前付けには時間がかかるし，同一人でも毎回（エまの結果）方針が一定しない。今日，端末に向い直接プログラムする機会も増えて好い加減な名前付けで済ましかねない。完全は無理でも実用的なレベルで，品質良い名前付けを自動化しそれを蓄積できれば，プログラマや読み手の負担が軽減でき，ソフトウェアの生産性向上に役立つと期待できる。

4. NameMasterの役割　— SymbolicNameとNameの対応管理，識別子の算出

ユーザの使うSymbolicNameとマシンの使うNameの対応を1:1に保つことを保障する機構を備えたSymbolicNameのデータベースマネージャである。加えて、状況に応じて的確かつ体系的な識別子を計算するサービスを提供する。例えば、あるプリンタ状態表示ウィンドウに於いてはPrintDeviceBusyより単にBusyで十分であり望ましい。短いので一見で見てとれるからである。*

* 省略形や造語の発音がまちまちだと，不便なことは日常体験する所である。蛇足ながら，DinnerBellではコメント文は分布や品質の管理不能ゆえに廃止し、識別子の自己記述で置換える。/* Comment statement considered harmful */

* 加えて，CRT画面が大きくないので圧縮された表現が好ましい。ObjectPeeperの圧縮技法をNameMasterによって実装する予定[3]。

5. NameMasterを用いる利点 ― 幅広いサービスの提供

NameMasterが統括する範囲は1ユーザとする。なるべく広いほうがよい識別子から何を連想するかに個人差があるからである。この様に名前付けを一のプログラムを越えて適用することは、自動化して初めて可能になる。的確な名前がいつでも同じ物を意味するので、マニュアルなどを引く回数が減るなどがスムースに運ぶことが期待される。

さらに、体系的な名前付けにより名前同志の類似性がそれらの指すの類似性を反映しているので、名前間の類似度を決めればクラスタ分析など手法を適用し、対象を分類することができる。モジュールをこの方法で大分類した後、キーネームを鍵として詳細検索する応用が考えられる。

6. NameMasterの使用の強制 ― 拘束を嫌うプログラマ対処法

プログラマは他からの強制を嫌うので工夫がいる。モジュールを格納するライブラリを2つにする。NameMasterが管理するモジュールライブラリとしない一般ライブラリである。前者には前節のような種々のサービスが提供サービスを利用しないユーザにモジュールライブラリの利用を促す。NameMasの方針を守る良いプログラマにとっては負担が小さい。

7. NameMasterの使用法 ― シナリオ

SymbolicName作成：ユーザがSymbolicNameの記述を専用エディタで作NameMasterが、名前のありふれ度[11]などを用いて、記述の検査をした後に納する。データベースには名前の一斉取換えなどの操作系が完備される。

Moduleライブラリ化：一般ライブラリからモジュールライブラリへモジールを移すこの際、元の識別子はNameMasterによって適切なSymbolicNameに置き換えられる。

8. SymbolicNameの記述 ― Objectに関する知識表現

知識工学で確立した知識表現を使ってObjectの意味記述を行う。識別の計算は少なくとも自然言語より簡単である。

9. プログラムに出現する名前の性質

知識化するためには、実際の名前について調べる必要がある。例えば、右図はあるプログラム群に於ける名前の出現頻度の降順グラフである。両log軸グラフで傾きが-1に近く、経験則であるZiphの法則に従うと言える[11]。これは、プログラマが使う名前の構成が書き言葉の構成と相似であり、比較的小数の単語が名前として繰り返し現われることを示唆している。調べた範囲ではZiphの法則はよく成り立っていた。